Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Applied Logic 4 (2006) 192–214
JOURNAL OF APPLIED LOGIC
www.elsevier.com/locate/jal

Efficient spatio-temporal data mining with GenSpace graphs

Howard J. Hamilton *, Liqiang Geng, Leah Findlater, Dee Jay Randall

Department of Computer Science, University of Regina, Regina, SK, Canada S4S 0A2

Available online 22 July 2005

Abstract

We describe a method for spatio-temporal data mining based on GenSpace graphs. Using familiar calendar and geographical concepts, such as workdays, weeks, climatic regions, and countries, spatio-temporal data can be aggregated into summaries in many ways. We automatically search for a summary with a distribution that is anomalous, i.e., far from user expectations. We repeatedly ranking possible summaries according to current expectations, and then allow the user to adjust these expectations. We also choose a propagation path in the GenSpace subgraph that reduces the storage and time costs of the mining process.

Keywords: Data mining; Knowledge discovery; Spatio-temporal data mining; Spatial data mining; Temporal data mining; Summarization; Domain generalization graphs; GenSpace graphs

1. Introduction

Recently, spatio-temporal data mining has been identified as a distinct research area concerned with knowledge discovery from datasets containing explicit or implicit temporal, spatial or spatio-temporal information [17]. People have rich background knowledge about time and space, including vivid knowledge of multiple ways that time and space

* Corresponding author.
E-mail addresses: hamilton@cs.uregina.ca (H.J. Hamilton), gengl@cs.uregina.ca (L. Geng).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2005.06.005

values can be aggregated. A *calendar attribute* has a domain consisting of date and time values, such as birth dates or check out times [21], and a *geospatial attribute* has a domain consisting of Earth-based location values, such as geographic coordinates or city names. Strong effects on patterns in data may occur because calendar values reflect strong physical constraints (e.g., Earth's rotation and revolution around the Sun) and strong cultural constraints (e.g., month-end paydays, Christmas shopping). Similarly, geospatial values also reflect strong physical constraints (e.g., proximity, weather) and strong cultural constraints (e.g., political regions, urban versus rural).

In this paper, which extends our previous report [11], we describe a method for finding interesting summaries regarding calendar and geospatial attributes at various concept levels. By *summarization*, we refer to the formation of interesting, compact descriptions of data. Summarization was listed by Fayyad et al. as one of the six primary data mining tasks [9]. Early work on attribute-oriented induction [6], function finding [23], multivariate visualization techniques, and derivation of summary rules provided diverse approaches to summarization online analytical processing (OLAP), data cubes with rollup and drilldown operators, and the *rollup* and *cube by* operators in SQL all address the task of summarization.

We have found three chief weaknesses in the previously described approaches to summarization. It is common to create numerous summaries, all valid, for the same data. Unfortunately, it is tedious and time consuming for the data analyst to have to examine these summaries one by one to assess them. Secondly, incorporating domain knowledge into the summarization process is not facilitated by some of these approaches. Attribute-oriented induction does allow background knowledge to be incorporated in the form of a concept hierarchy for each attribute, but it does not provide a means of specifying multiple ways of aggregating the values for an attribute during a single data mining task. Lastly, these methods provide limited scope to handle the changes in the user's knowledge that naturally occur during knowledge discovery. Consider an example. Suppose that it is discovered and reported to the user that purchasers in Canadian cities buy fewer basketball-related goods than those in US cities by analysing information from North American sporting goods stores. Then, it will be subsequently less interesting to discover that purchasers in Saskatchewan (a province in Canada) buy fewer basketball-related goods than those in other parts of North America. In the application of our DGG-Discover software to a wide variety of commercial and institutional databases over the past five years, we have found that these limitations are most manifest for temporal and spatial attributes.

Our summarization method is based on GenSpace graphs, which are formed by combining domain generalization graphs that have been extended by information about expectations. A *domain generalization graph* (DGG) [12,15,16] is a graphical structure that can be used both as a navigational aid by the user and as a guide to heuristic data mining procedures. Each path in the graph corresponds to a generalization consistent with the specified generalization relations. An *expected distribution domain generalization* (*or ExGen*) *graph* is a DGG where each node has been augmented with a description of the expected distribution of the values in the corresponding domain. Initial and changing domain knowledge can be described by ExGen graphs. For a calendar attribute, we created a standard ExGen graph that describes all well-known temporal relationships, including the number of days in a week, month, and year, as well as the seasons and leap years. To customize the calendar

ExGen graph for a particular application, additional information about workdays, financial quarters, academic terms, etc. can be added. During the knowledge discovery process, the expectations at a particular node, which reflect a user's knowledge about the corresponding domain, can be updated. As well, expectations can be propagated to other nodes in the graph; for example, if the user revises the expectation about the fraction of movies watched in the evening, then the expectation about the fraction of movies watched from 8:00 pm to 9:00 pm can be automatically adjusted.

For a calendar attribute, our approach has five steps. First, a domain generalization graph for a calendar attribute is created by explicitly identifying the domains appropriate to the relevant levels of temporal granularity and the mappings between the values in these domains. Second, a probability distribution is associated with each node in the graph. Third, the data are aggregated in all possible ways consistent with this graph. Aggregation is performed by transforming values in one domain to another, according to the directed arcs in the domain generalization graph. Each aggregation is called a *summary*. Fourth, the summaries are ranked according to their distance from the expected distribution for the appropriate domain, using a diversity-based interestingness measure [14,15]. Fifth, the highest ranked summaries, i.e., the summaries whose observed distributions are the farthest from expectations, are displayed. Expectations are then adjusted and steps repeated as necessary.

Our work can be contrasted with recent research on the connection between multiple temporal granularities and data mining. Granularity factors that affect data mining were described by Andrusiewicz and Orlowska [1]. Bettini et al. provide conventions similar to those given here for naming the multiple temporal granularities, although they do not provide a data structure similar to DGGs for representing the relationships between these granularities or for recording or manipulating expectations [5]. They define a *calendar algebra* to represent granularities of calendar data and the relationships between those granularities. Specified *calendar operations* are used to create new granularities, either by recursively grouping data from an existing granularity or by filtering data from a granularity. They apply their system in the context of data mining by looking for frequent event sequences that have a specified minimum confidence at some level of temporal granularity. Bertino et al. build on the work of Bettini et al. to specify the syntax and semantics of expressions involving data with multiple temporal granularities [4]. Combi et al. use a much simpler structure than our calendar DGG with an ordered granularity from SUP (top) to year, month, day, hour, minute, second, INF (bottom) [7,10]. However, they cannot handle expectations or phenomenon such as weeks.

In data mining, work on contrast sets has identified pairs of values that lead to significantly different outcomes in the context of particular combinations of values [3], but this approach does not allow known generalization relations among attributes or user expectations to be incorporated. Work on temporal data mining has emphasized searching for recurring patterns in time series [2]; our method is not restricted to time series. Adding temporal semantics to association rules has also attracted attention [18–20]. Rainsford and Roddick provide a method based on structured relationships among temporal relations, rather than our structure among domains [20]. Li et al. build on the Apriori algorithm for mining association rules to include temporal semantics [19]. Compared to all these methods, our method has three distinctive features: (1) it provides a more expressive structure

to represent the generalization relation for spatio-temporal attributes; (2) it incorporates the user's knowledge by explicitly representing his or her expectations; and (3) it finds the most interesting summaries interactively, which refines the user's expectations step by step.

The remainder of this paper is organized as follows. In the following section, we review domain generalization graphs and present a particular graph for a calendar attribute. We also describe ExGen graphs and GenSpace graphs, and explain how expected distributions are attached and propagated. In Section 3, we identify efficient propagation paths in GenSpace graphs. In Section 4, we briefly describe our methodology, illustrate its application to two data sets, and report the results of experimenting on propagation efficiency. Finally, in Section 5, we present our conclusions.

2. Generalizing calendar data with DGG and GenSpace graphs

Informally, a DGG can be thought of as a graph showing possible generalizations as paths through a graph. Fig. 1 shows part of a DGG for a calendar attribute (simplified from [21]). The node labelled *YYYYMMDDhhmm* represents the most specific domain considered, i.e., the finest granularity of our calendar domain is one minute. Higher-level nodes represent generalizations of this domain. Each link in the DGG represents a generalization relation. For example, a link pointing from *decade* to *century* means that values in the *century* node can be generalized from values in the *decade* node. The *YYYYMMDDhhmm* node represents the finest domain of calendar values in the DGG, and it is called the *bottom node*. The *Any* node represents the most generalized domain of calendar values, which includes only the value *Any*. This node is called the *top node*. To handle data with calendar values specified to finer granularity, e.g., seconds, more specific nodes could be added to the DGG.

Formally, a domain generalization graph is defined in terms of a generalization relation (adapted from [15,16]). Given a set $X = \{x_1, x_2, \ldots, x_n\}$ representing the base-level domain of some attribute and a set $P = \{P_1, P_2, \ldots, P_m\}$ of partitions of the set X, we define a nonempty binary relation $\preceq$ (called a *generalization relation*) on P, where we say $P_i \preceq P_j$ if for every section $S_a \in P_i$, there exists a section $S_b \in P_j$, such that $S_a \subseteq S_b$. A *section* is an element of a partition. It is a labelled subset of the domain of the bottom node. It can be defined arbitrarily by the user as long as it has a logical meaning to him or her, and is not necessarily connected in terms of time and space for each attribute. The generalization relation $\preceq$ is a partial order relation. If $P_i \preceq P_j$, for each section $S_b \in P_j$, there exists a set of sections $\{S_{a_1}, \ldots, S_{a_k}\} \subseteq P_i$, denoted $Spec(S_b, P_i)$, such that $S_b = \bigcup_{i=1}^{k} S_{a_i}$.

In Fig. 1, the sections in *YYYYMMDD* could include 2001/01/01, 2001/01/02, …, and 2003/12/31. The sections in *day of week* are *Sunday*, *Monday*, …, and *Saturday*. The *Saturday* section is a labeled subset of the domain in the bottom node, *YYYYMMDDhhmm* (representing time values at the granularity of minutes). Each specific value from the bottom node that is within any Saturday is grouped into the *Saturday* section in the *day of week* node. The *YYYYMMDD* node (representing time values at the level of a specific date) has a generalization relation with the *day of week* node, because all specific values from the bottom node that are in a single section in the *YYYYMMDD* node, say 2003/04/06, are

contained in the same section in the *day of week* node. Values in a section of *day of week* are not necessarily adjacent. For example, 2003/03/10 01:00am and 2003/03/17 01:00am are elements in the *Monday* section, while 2003/03/11 01:00am, which between those two time points, does not belong to *Monday*.

Definition 1. A *domain generalization graph* (*DGG*) $G = \langle P, E \rangle$ is constructed based on a generalization relation $\langle P, \preceq \rangle$ as follows. The nodes of the graph are the elements of P. There is a directed arc from P_i to P_j iff $P_i \neq P_j$, $P_i \preceq P_j$, and there is no $P_k \in P$ such that $P_i \preceq P_k$ and $P_k \preceq P_j$. Each node corresponds to a domain of values. Each arc corresponds to a generalization relation, which is a mapping from the values in the domain of the initial (or parent) node to that of the final node (or child) of the arc. The *bottom* (or source) node of the graph corresponds to the original domain of values X and the *top* (or sink) node T corresponds to the most general domain of values, which contains only the value ANY.

The calendar DGG in Fig. 1 can be used to guide the generalization of calendar data into higher-level concepts. For example, we generalize from *YYYYMMDD* to *YYYYMM* by removing the *DD* information from the calendar attribute. When a new representation is required in the calendar domain, this DGG can be extended by adding new nodes and arcs and by defining new generalization relations associated with the arcs.

Four types of generalization relations are associated with the arcs in a calendar DGG: granularity, subset, lookup, and algorithmic. For *granularity generalization*, we assume

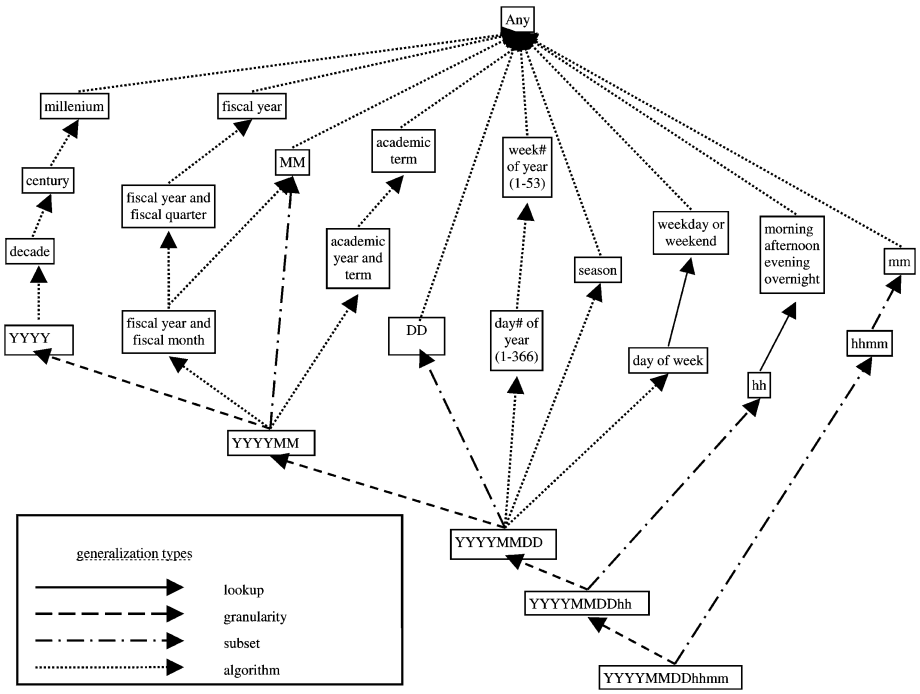

Fig. 1. Simplified domain generalization graph for a calendar attribute.

that *YYYYMMDDhhmm* can be represented as five subattributes (*YYYY*, *MM*, *DD*, *hh*, *mm*). We generalize by suppressing subattributes from least significant (*mm*) to most (*YYYY*). All four domains this creates are shown in Fig. 1. For example, granularity generalization could be used to generalize from *YYYYMMDDhhmm* to *YYYYMMDD*. *Subset generalization*, which includes granularity generalization as a special case, discards any combination of subattributes. The remaining subattributes need not be adjacent. For example, we could generalize from *YYYYMMDDhhmm* to *MMhh*. Fig. 1 shows only a few of the domains that can be created by subset generalization. *Lookup generalization* uses a lookup table to generalize from a lower-level node, such as *day of week* with partition [*Sunday*, *Monday*, *Tuesday*, *Wednesday*, *Thursday*, *Friday*, *Saturday*], to a higher level node, such as *weekday or weekend* with partition [*Weekday*, *Weekend*]. *Algorithmic generalization* uses an algorithm to generalize. It allows convenient calculation of regular relationships, such as those required to determine the number of days in a year and the beginnings of seasons. Although algorithmic generalization subsumes the three previous types, we find the distinction useful and only refer to algorithmic generalization when no other is applicable.

An *expected distribution domain generalization* (*ExGen*) *graph* is a DGG that has a probability distribution associated with every node. Each distribution represents the expected probability of occurrence of the values in the domain corresponding to the node. For a node (i.e., partition) $P_j = \{S_1, \ldots, S_k\}$, we have $0 \leqslant \Pr(S_i) \leqslant 1$ and $\sum_{i=1}^{k} \Pr(S_i) = 1$, where $\Pr(S_i)$ denotes the probability of occurrence of a value $S_i \in P_j$, i.e., the ith section in node P_j. Each probability distribution represents the user's expectation for the frequency of occurrence of the values in the domain corresponding to the node. For example, if the domain is the names of countries of the world and expectations are based on population, the distribution could be specified by giving each country's name associated with the ratio of that country's population to the world population. Such a distribution is most often called a *prior* in statistics.

The simplest approach is to assume uniform distribution for all domains. Unfortunately, this approach may suggest inconsistent distributions. As a simple example, uniform distribution over the *day of week* domain ($1/7$ for each day) is inconsistent with uniform distribution over the *weekday or weekend* domain ($1/2$ for the value Weekday and $1/2$ for the value Weekend). Two days, Saturday and Sunday, with a total expectation of $2/7$ are generalized to Weekend, with a total expectation of $1/2$, which is inconsistent.

To define consistency, assume node Q is a parent of node R in an ExGen graph, and therefore for each section $S_b \in R$, there exists a set of sections $Spec(S_b, Q) = \{S_{a_1}, \ldots, S_{a_k}\} \subseteq Q$, such that $S_b = \bigcup_{i=1}^{k} S_{a_i}$. If for all $S_b \in R$, $\Pr(S_b) = \sum_{i=1}^{k} \Pr(S_{a_i})$, we say that Q and R are *consistent*. In an ExGen graph, we say that node R is *bottom-consistent*, i.e., consistent with the bottom node X, if for all $S_i \in R$, $\Pr(S_i) = \sum_{x \in S_i} \Pr(x)$. We say an ExGen graph G is *consistent* if all pairs of adjacent nodes in G are consistent.

Lemma 1. *An ExGen graph G is consistent iff every node in G is bottom-consistent.*

Proof sketch. Proof follows by induction based on the distance from the bottom node and the expectations for each section S_b at node R equaling the sum of the expectations of the more specific values at node Q that correspond to this section, namely $Spec(S_b, Q)$. □

To avoid inconsistencies and simplify the process of specifying expectations, a distribution can be specified for the bottom node, and then propagated upward to all nodes (*bottom-up propagation*). Or a distribution can be associated with a single node in the ExGen graph and then using an assumption of a uniform (or other) distribution among the values in each section, it can be propagated to the bottom node of the graph, and bottom-up from there.

Lemma 2. *If an ExGen graph G is constructed by bottom-up propagation from a DGG D and a distribution E for the values in X, the bottom node of the graph, then graph G is consistent.*

Proof sketch. Proof follows by induction based on the distance from the bottom node, as with Lemma 1. □

For example, if the logins to a system are expected to be uniformly distributed among all days in a three week period, then propagating upward gives a uniform distribution for the seven members of *day of week*, and propagating further upwards gives a $\{2/7, 5/7\} = \{0.29, 0.71\}$ distribution for the two members of the *weekday or weekend node*. (We use two significant digits in this paper, but double precision in our implementation.)

To guarantee that propagation is consistent, we first propagate the new expectations specified by user directly to the bottom node (if expectations in more than one nodes are changed, we propagate them to the greatest lower bound node, and then propagate them to the bottom node), and then propagate the expectations up to the entire graph. In this manner, the entire graph is consistent, according to Lemma 2.

When several attributes have ExGen graphs, aggregation is performed to all nodes in the Cartesian product of the ExGen graphs to form a GenSpace graph. By the *Cartesian product of the ExGen graphs*, we mean every combination of nodes from the ExGen graphs where one node is taken from each ExGen graph. The expectation of a combination of values (or *generalized tuple*) $t = (d_1, d_2, \ldots, d_m)$ is by default the product of the expectations of the values of its component attributes; i.e., $e(t) = e(d_1)e(d_2)\ldots e(d_m)$. (If an expectation has been specified for a subset of the attributes, as a joint-probability distribution, then value is derived from this distribution instead of the product of the expectations.)

During the data mining process, an output summary can be produced for every combination of nodes in the ExGen graphs where one node is taken from each ExGen graph. A summary is produced as a file of comma-separated values, which can be readily displayed and processed with Microsoft Excel and other standard tools. Given a set of summaries, an interest measure assigns a numeric score to each. These scores can be used to rank the results and determine which generalized relations are consistent with an expectation and which ones conflict with it. An interest measure is computed by comparing an observed distribution to an expected distribution.

3. Finding efficient propagation paths in GenSpace graphs

After the framework of a GenSpace graph has been generated, the user can mark some nodes as uninteresting before the propagation process starts. Usually, the uninteresting

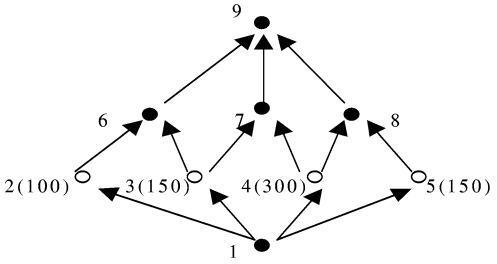

Fig. 2. Example 4.

nodes are in the lower levels of the GenSpace graph, because these nodes correspond to larger summary tables, which are harder to comprehend. Pruning even a small percentage of such nodes will significantly reduce storage costs. We can prune an uninteresting node by connecting all its parent nodes to its child nodes, but this will destroy the one-step-generalization feature of a link. To preserve this feature, some uninteresting nodes must not be pruned. Our problem is to find the uninteresting nodes that should be preserved.

In Fig. 2, the white ovals (nodes N_2, N_3, N_4, and N_5) denote the uninteresting nodes and the black ovals denote the potentially interesting nodes. The numbers in parentheses denote the space cost of the nodes. We intend to prune a subset of uninteresting nodes such that all potentially interesting nodes can be reached from the bottom node. We can prune N_3 and N_5, or N_2 and N_5, or N_2 and N_4 to satisfy this constraint. Here we prefer to prune N_2 and N_4, because these two nodes require the most storage units ($100 + 300 = 400$).

Before we give formal description of the problem, we first give some definitions.

Definition 2. If a non-uninteresting node only has uninteresting nodes as its parents, we call it an *endangered node*.

If pruning is not properly performed, these nodes might not obtain expectations from bottom up propagations.

Definition 3. The parent nodes of endangered nodes are called *candidate nodes*.

We need to select a subset of the candidate nodes to prune and preserve the rest in the GenSpace graph, although we will hide them from the user.

Definition 4. A candidate node's endangered children are called the *cover of the candidate node*. The *node preservation problem* is to find a subset of the candidate nodes such that the union of their covers equals the set of endangered nodes and the storage cost of these nodes is minimum.

In Fig. 2, the endangered node set is $\{N_6, N_7, N_8\}$, the candidate node set is $\{N_2, N_3, N_4, N_5\}$. $\text{Cover}_{\text{N2}} = \{N_6\}$, $\text{Cover}_{\text{N3}} = \{N_6, N_7\}$, $\text{Cover}_{\text{N4}} = \{N_7, N_8\}$, and $\text{Cover}_{\text{N5}} = \{N_8\}$. We represent these facts in Table 1.

In Table 1, each row describes a candidate node, and each column describes an endangered node. If a candidate is a parent of an endangered node, we put the storage cost in

Table 1
Cover sets

Candidate	Endangered		
	N_6	N_7	N_8
N_2	100		
N_3	150	150	
N_4		300	300
N_5			150

Function SelectCandidateNodesOneStep

1. Search the GenSpace graph and find all the endangered nodes.
2. Find the set of candidate nodes corresponding to the endangered nodes.
3. Construct the coverage relation table (as in Table 1).
4. While the set of the endangered node set is not empty,
   4.1 Select a candidate node with the minimum storage coverage ratio. If there is a tie, select one with greater coverage. If there is a tie again, we randomly select one.
   4.2 Eliminate the covered endangered nodes from the endangered nodes to eliminate the selected candidate node from the candidate node set.
   4.3 Recalculate the coverage and storage coverage ratio for the left candidate nodes.
5. Check the selected node set one by one in the reverse order of selection using forward adding backward elimination strategy and eliminate the redundant nodes.

Fig. 3. Function SelectCandidateNodesOneStep.

the corresponding cell. Here $\{N_3, N_5\}$ is a subset that covers the entire set of endangered nodes, and its storage cost $(150 + 150 = 300)$ is minimum. Therefore, we choose to keep N_3 and N_5 and prune nodes N_2 and N_4.

We use a greedy heuristic to obtain an approximate solution to the node preservation problem. Since we want to choose the nodes that have smaller storage and greater coverage, we define the storage coverage ratio $\mathrm{SCR}(N) = \mathrm{Storage}(N)/\mathrm{Coverage}(N)$ for each candidate node N. At each step, we select a node with the lowest storage coverage ratio to preserve. After the selection, we remove the endangered nodes that are covered by the selected node, i.e., we delete their corresponding columns in the table, and we also delete the row in the table corresponding to the selected node. Then we recalculate the storage coverage ratios in the new table and repeat the selection process. This process continues until all columns of the table have been deleted, i.e., all the endangered nodes are covered. After obtaining the subset of nodes to preserve, we use backward elimination to eliminate any redundant nodes in this subset. The algorithm is presented in Fig. 3.

Let m and n denote the numbers of the candidate and endangered nodes, respectively. The worst case occurs when only one more endangered node is covered each time we select a candidate node. If $m < n$, the worst case complexity $\mathrm{O}(m^2)$. If $m \geqslant n$, the worst case complexity for this algorithm $\mathrm{O}(m(m-n))$.

In Example 5, because node N_3 has the minimum storage coverage ratio $(150/2 = 75)$, we select it and eliminate nodes N_6 and N_7 of the endangered node set. Next, we choose N_5, because it has the minimum storage coverage ratio 150. (Initially, the storage coverage

Algorithm SelectCandidateNodes

1. Create the set of endangered nodes by scanning the GenSpace graph.
2. While the set of endangered nodes is not empty,
   2.1 Find the nodes to preserve using Function *SelectCandidateNodesOneStep*.
   2.2 Find the set of endangered nodes in the selected node set.

Fig. 4. Function SelectCandidateNodes.

ratio of N_4 was 150, but after we eliminated N_7, the ratio became 300.) At this point, all endangered nodes have been covered. So we keep N_3 and N_5 and prune N_2 and N_4.

After we select a new uninteresting node to preserve, it can become an endangered node again. We have to guarantee that all the newly selected candidate nodes are safe for propagation. Fig. 4 gives the algorithm for selecting candidate nodes.

4. Method and sample applications

4.1. Applications

We now describe the data mining technique we implemented in our DGG-Discover 5.0 software. For clarity, the method is first briefly explained and then its application to two datasets is described. The first dataset concerns weather in the province of Saskatchewan, Canada, and the second concerns logins to a shared computer system called Hercules at the University of Regina.

4.1.1. Saskatchewan weather data

Our goal was to assess whether, with no previous experience with this dataset, this discovery methodology could guide exploration of the data. We used the daily high temperature (in 0.1 degree Celcius) and daily total precipitation (in mm, with snow converted to equivalent water) for all weather stations for all days from January 1, 1900 to December 31, 1949. Other fields concerning low temperatures and snowfall were not used in our analysis. The number of daily weather observations (tuples) was 211,534. Example data is given in Table 2, where StationName and Latitude and Longitude depend on the Station attribute.

The attributes we used in our experiment are *Station, Time*, *HighTemperature* (temperature in Celsius), and *TotalPrecip* (precipitation in mm). Attribute *Time* has format *YYYYMMDDHHMMSS*, including the information of year, month, day, hour, minute, and second. We generalize it into 9 nodes: *YYYYMMDDHHMM* (minute), *YYYYMMDDHH* (hour), *YYYYMMDD* (day), *YYYYMM* (year and month), *MM* (month), *YYYY* (year), Decade, Season, and *Any* (any time). *HighTemperature* was generalized into three nodes *TempRange*, *TempSplit*, and *Any*, *TotalPrecip* was generalized into three nodes *PrepRange*, *PrepSplit*, and *Any*, and *Station* was generalized into four nodes according to geographic features and node *Any*. Fig. 5 shows the DGGs for these attributes. The three paths in the Station DGG correspond to the three maps shown in Fig. 6. In the GenSpace graph, there are $(5+1)*(3+1)*(3+1)*(4+1) = 480$ nodes (479 summaries).

Table 2
Sample weather data

Station	StationName	Latitude	Longitude	Date	High temperature	TotalPrecip
4012400	Estevan A	49.217	102.967	12/27/1944	−12.8	0
4063560	Island Falls	55.533	102.350	12/27/1944	−22.2	0
4016560	Regina A	50.433	104.667	12/27/1944	−14.4	0
4018160	Tugaske	50.883	106.300	12/27/1944	−20.6	0
4048520	Waseca	53.133	109.400	12/27/1944	−18.9	0
4019080	Yorkton	51.267	102.467	12/27/1944	−18.9	0

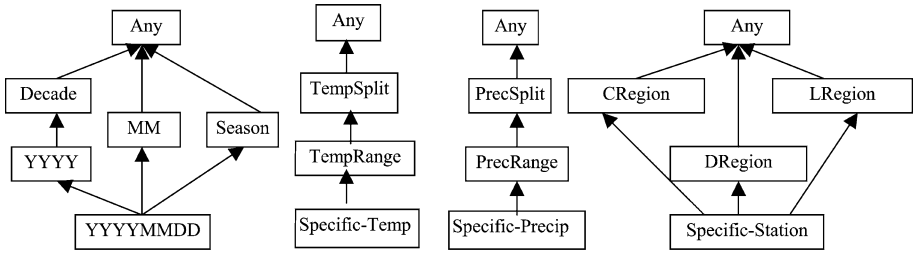

Fig. 5. DGGs for Date, HighTemperature, TotalPrecip, and Station Attributes.

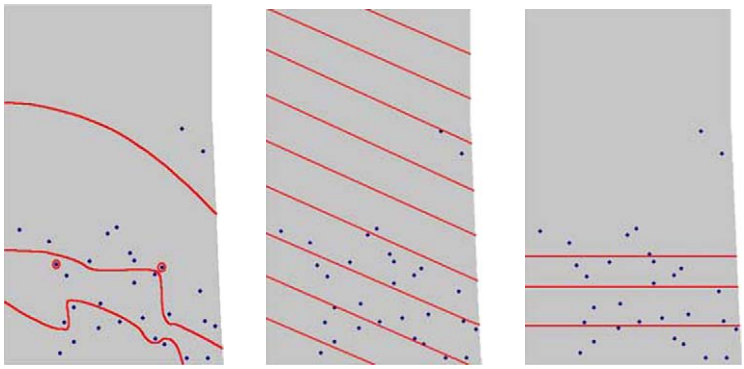

Fig. 6. Maps corresponding to the interior DGG nodes for the station attribute. (a) Clustered regions (CRegion). (b) Distance regions (DRegion). (c) Latitude regions (LRegion).

Fig. 6(a) shows the stations clustered using the k-means algorithm with $k = 6$ (other values of k gave less plausible maps). Fig. 6(b) shows ten regions defined based on the adjusted distance d to any point (Lat, Long) from the southwest corner of the province, which is at (49°N, 110°E), using the formula

$$d = (\text{Lat} - 49) + 0.35(110 - \text{Long}).$$

This assumes that similar weather occurs along a slanted line across the province. The 0.35 is an arbitrary constant defined based on a map of the province showing ecological regions [8]. Since the northeast corner of the province is at (60°N, 102°E), the values for d ranged from 0 to 13.8. To create the DistRange node, we divided this range into 10 equal-sized intervals. Fig. 6(c) shows the stations grouped from south to north into four regions (South, LowMid, HighMid, and North) to create the LRegion node.

Associating expectations. In the beginning of the exploration process, we assumed initial expectations for some nodes in the DGGs. For the *Date* attribute, we assumed a uniform distribution at the *Year* (*YYYY*) node, i.e., we assumed an equal number of observations from each year 1900–1949. For the *Station* attribute, we assumed a uniform distribution at each weather station (*Specific-Station* node). For the *HighTemperature* attribute, we assumed a uniform distribution at the *TempRange* node, which means that cold days, cool days, warm days, and hot days all have probability value of 0.25. Similarly, for *TotalPrecip*, we assumed a uniform distribution at the *PrecRange* node, which means that the numbers of days for no precipitation, low precipitation, medium precipitation and high precipitation are equal. Initially, we also assumed that the four attributes are independent, therefore, we were able to generate GenSpace graph with initial expectations.

Generalize the data. Next the weather data are aggregated in all possible ways consistent with the domain generalization graphs. Since the *Date*, *HighTemperature*, *TotalPrecip*, and *Station* DGGs have 6, 4, 4, and 5 nodes, respectively, the number of aggregations or summaries produced is $6(4)(4)(5) = 480$. Each summary is evaluated with an interestingness measure and also optionally stored as a Comma-Separated-Value (CSV) file.

Rank the summaries. The summaries are ranked according to their distance from the expected distribution using a selected interestingness measure. Table 3 shows the top 10 summaries for the first run. To get more insight into the highest ranked summaries, we use the chi-square test to test the fitness of the observations and expectations and determine which detailed summary records should be provided to the user. In this case, the highest ranked summary, which corresponds to the (Any, Any, Any, PrecSplit) node combination, is automatically translated into the English sentence "More readings (79%) have precipitation = NONE than expected (50%)". This summary shows that, when Station, Date, and High Temperature are ignored (set to Any), the percentage of daily observations in the data with rain is 21%, and the number without is 79%. Since the original expectation was 50%/50%, the actual distribution is far from the expected one, and according to the variance measure, farthest from the expectation of any node combination.

Adjust expectations. At this point, the user has learned something about the domain, and the expectations can be adjusted according to the acquired knowledge. To continue this example, we assume that the distribution is simply accepted for the (Any, Any, Any, PrecipSplit) node and propagated to all the other nodes in the GenSpce graph. This assumption follows in a straightforward fashion from the results, and can be readily automated. The effect on further data mining corresponds to saying: "I accept that only 21% of the days have precipitation; now, don't tell me about that again or about any logical consequence of that".

Table 3
Top-ranked summaries after run 1

Station	Date	Temperature	Precipitation	Variance
Any	Any	Any	PrecSplit	0.6884
Any	Any	TempSplit	PrecSplit	0.1166
Any	Any	Any	PrecRange	0.1104
LRegion	Any	Any	PrecSplit	0.0263
Any	Any	TempRange	PrecSplit	0.0255
Any	Any	TempSplit	PrecRange	0.0251
Any	Season	Any	PrecSplit	0.0247
CRegion	Any	Any	PrecSplit	0.0207
DRegion	Any	Any	PrecSplit	0.0202
Any	Decade	Any	PrecSplit	0.0181

Continue data mining. After propagating the revised expectations and calculating the interestingness measures of the summaries, we obtain the 10 most interesting summaries in Table 4. Most summaries with *PrecipSplit* or *PrecipRange* have disappeared from the top ten list because the change in the expectations of *PrecipSplit* affects that of *PrecipRange* appropriately. The summaries (Any, Any, Any, PrecRange) and (Any, Decade, Any, PrecSplit) remain in the top ten. However, the summary (Any, Any, Any, PrecRange) ranks 7th instead of 3rd, and its interestingness measure decreased from 0.110 to 0.003. Although summary (Any, Decade, Any, PrecSplit) has a higher ranking (from 10 to 8), its interestingness measure decreased from 0.018 to 0.003. This indicates that through propagation from node (Any, Any, Any, PrecSplit), the distribution for other nodes in this GenSpace graph has been adjusted appropriately.

Now, the highest ranked summary tells the user that the expectation for the *Decade* node is far away from the observed one. Among the five decades from 1900 to 1949, the percentage of observations from the decades in order are: 7%, 13%, 21%, 25%, and 34%. Again, the user can simply accept this, or explore deeper to understand why. The actual reason was that only a few weather stations existed in 1900 and others were gradually added. This relationship is best addressed by creating a joint distribution (discussed further below) between Station and Date (at the Year or YYYYMMDD node). For this example, we assume that the user simply accepts the observed distribution as the expected for Decade and propagates it throughout the entire GenSpace graph.

Tables 5–8 list the ten most interesting summaries for runs 3 to 6.

When the top-ranked summary has more than one domain value that is not "Any", the summary corresponds to a *joint probability distribution* (or *joint expectation*). For example, after run 4, the top-ranked summary is (Any, Season, TempSplit, Any), which means that the proportion of Low Temperature and High Temperature days varies with the season. To accept this distribution, a joint expectation between Season and TempSplit is created. All subsequent runs will consult this table whenever an expectation for the combination of Season and TempSplit needs to be calculated. After propagation of the joint expectation and calculation of the interestingness of the summary, we can see that not only has (Any, Season, TempSplit, Any) disappeared from the top ten summaries, but also closely related nodes (Any, Season, TempSplit, PrecSplit) and (Any, Season, TempRange, Any) have become less interesting and also disappeared.

Table 4
Top ranked summaries after run 2

Station	Date	Temperature	Precipitation	Variance
Any	Decade	Any	Any	0.0105
Any	Any	TempSplit	Any	0.0084
Any	Season	TempSplit	Any	0.0067
LRegion	Any	Any	Any	0.0063
CRegion	Any	Any	Any	0.0058
Any	Any	TempRange	Any	0.0037
Any	Any	Any	PrecRange	0.0029
Any	Decade	Any	PrecSplit	0.0028
Any	Decade	TempSplit	Any	0.0025
DRegion	Any	Any	Any	0.0023

Table 5
Top ranked summaries after run 3

Station	Date	Temperature	Precipitation	Variance
Any	Any	TempSplit	Any	0.0084
Any	Season	TempSplit	Any	0.0067
LRegion	Any	Any	Any	0.0063
CRegion	Any	Any	Any	0.0058
Any	Any	TempRange	Any	0.0037
Any	Any	Any	PrecRange	0.0029
DRegion	Any	Any	Any	0.0023
Any	Season	TempSplit	PrecSplit	0.0021
Any	Season	TempRange	Any	0.0020
Any	Any	TempSplit	PrecSplit	0.0019

The results that we obtain from the system are an ordered list of summaries that have high interestingness values. Within each summary is a list of expected and observed probability distributions. If the variance of a record in the summary is greater than a threshold, we will highlight this record and provide it to the user. If the observed probability is higher than the expectation, we say that this record is more likely than what the user expected and highlight it in red. Otherwise, we say it is less likely and highlight it in blue. For example, if the "summer, hot" record has observation and expectation probabilities of 0.2 and 0.1, respectively, we highlight it in red, which means that "hot, summer" records occurred more frequently than expected.

As the knowledge discovery process continues, other relationships continue to be found. To summarize, in Table 9, we list one relationship from each of several of the first 21 runs. Although the user may have originally expected to learn about the weather from this dataset, the actual exploration yielded a substantial amount of information about the dataset itself, including that summer readings were taken more faithfully than winter ones. Due to the large size of the dataset used, all relationships reported were statistically significant at the 0.001 significance level.

Table 6
Top ranked summaries after run 4

Station	Date	Temperature	Precipitation	Variance
Any	Season	TempSplit	Any	0.0064
LRegion	Any	Any	Any	0.0063
CRegion	Any	Any	Any	0.0058
Any	Any	Any	PrecRange	0.0029
CRegion	Any	Any	PrecSplit	0.0017
DRegion	Any	Any	Any	0.0023
Any	Any	TempRange	Any	0.0023
Any	Season	TempSplit	PrecSplit	0.0020
Any	Season	TempRange	Any	0.0020
LRegion	Any	Any	PrecSplit	0.0017

Table 7
Top ranked summaries after run 5

Station	Date	Temperature	Precipitation	Variance
LRegion	Any	Any	Any	0.0063
CRegion	Any	Any	Any	0.0058
Any	Any	Any	PrecRange	0.0029
DRegion	Any	Any	Any	0.0023
Any	Any	TempRange	Any	0.0023
CRegion	Any	Any	PrecSplit	0.0017
LRegion	Any	Any	PrecSplit	0.0017
LRegion	Any	TempSplit	Any	0.0015
CRegion	Any	TempSplit	Any	0.0014
CRegion	Any	Any	PrecRange	0.0007

Table 8
Top ranked summaries after run 6

Station	Date	Temperature	Precipitation	Variance
CRegion	Any	Any	Any	0.0039
Any	Any	Any	PrecRange	0.0029
Any	Any	TempRange	Any	0.0023
DRegion	Any	Any	Any	0.0013
CRegion	Any	Any	PrecSplit	0.0012
CRegion	Any	TempSplit	Any	0.0010
Any	Any	TempSplit	PrecRange	0.0006
Any	Any	TempRange	PrecSplit	0.0005
CRegion	Any	Any	PrecRange	0.0005
Any	Season	TempRange	Any	0.0005

4.1.2. Login data

To demonstrate the utility of our approach, we will mention the results of applying our DGG-Discover software to 523,253 lines of output from the Unix "`last`" program, which produces a single line for each user session, showing user id, login and logout times, and the duration of the session. The input data were collected over a period of somewhat more

Table 9
Sample English-language highlights from the first 21 runs

Run #	Highest ranked node	Sample English-language description of the top-ranked difference
1	Any-Any-Any-PrecSplit	More readings (79%) have precipitation = NONE than expected (50%).
2	Any-Decade-Any-Any	Fewer readings (7%) have decade = 1900–1909 than expected (20%).
3	Any-Any-TempSplit-Any	More readings (56%) have temperature = COOLER than expected (50%).
4	Any-Season-TempSplit-Any	More readings (24%) have season = SUMMER and temperature = WARMER than expected (11%).
11	Any-Month-TempSplit-Any	Fewer readings (0%) have date = JANUARY and temperature = WARMER than expected (4%).
13	Any-Season-Any-Any	More readings (28%) have date = SUMMER than expected (25%).
20	LRegion-Any-TempSplit-Any	More readings (9%) have station = LATITUDE-SOUTH and temperature = WARMER than expected (7%).
21	Any-Season-Any-PrecSplit	More readings (7%) have season = SUMMER and precipitation = SOME than expected (6%).

Table 10
Top ranked nodes for login data

USER	LOGINTIME	Variance
Group	Any	0.0341
Any	WDWE	0.0177
Group	WDWE	0.0124
Group	FiscalYear	0.0063
Group	AcademicYear	0.0052
Group	YYYY	0.0052
Any	AcademicYearAndTerm	0.0020
Any	AcademicYear	0.0015

than two years, from June 1998 to June 2000. We focus on the login times, which can easily be mapped to the *YYYYMMDDhhmm* node in the ExGen graph, and user-ids, which can be grouped into categories such as CS undergrad, CS grad, staff, etc. The problem is to find interesting summaries of the data at various levels of granularity. In [21], a simpler version of the same problem with far less data was considered.

Using uniform expectations for each group of users and each node in the calendar DGG, the summaries listed in Table 10 were ranked highest. Since (*Group*, *Any*) is ranked highest, the expectation that people from all groups would log in with approximately equal frequency was furthest from matching the data, according to the variance measure. Full details on this example are given in [22]. For brevity, we use "*WDWE*" to represent "*weekday or weekend*", as described in Section 2.

The three highest ranked nodes in Table 10 are related. Investigation revealed that the (*Group*, *Any*) relation is unusual because `csugrd` and `unknown` have far more logins than expected while every other group has far fewer. The (*Any*, *WDWE*) relation is unusual because `weekday` logins are higher than expected based on the number of days. As well,

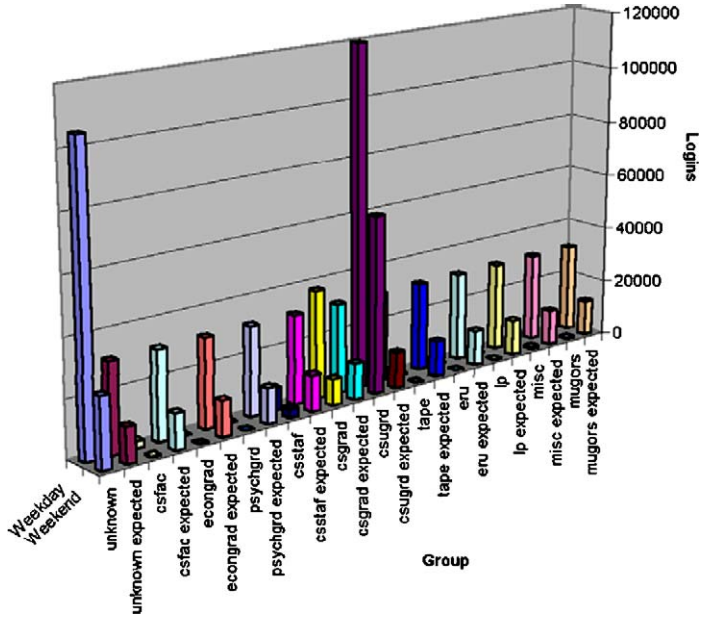

Fig. 7. Graph for (*Group*, *WDWE*) after the first run.

these two factors combine to make (*Group*, *WDWE*) unusual. This relationship is shown in Fig. 7 for (*Group*, *WDWE*). On the X axis (across), the first value is the observed number of logins for the first group, the second value is the expected number of logins for this group, the third value is the observed number for the second group, the fourth value is the expected number for the second group, and so forth. The `csugrd` group continues off the chart to just under 300000 logins on weekdays.

After the observed distribution was accepted as the expected distribution for the groups of users, the ranking of the nodes was as shown in Table 11. Some nodes, most notably (*Group*, *Any*) and (*Group*, *WDWE*), have lower rankings because of the added knowledge about the distribution in logins among the groups. The top ranked summary is (*Any*, *WDWE*), which tells us that the number of logins on a weekday is higher than on a weekend day. (*Any*, *AcademicYearAndTerm*) tells us that fewer logins occur during some particular terms (e.g., 1998-2) than during other terms, where there are three terms per year.

When variance is used as the measure of interest, our technique provides an easy way for a user to construct a hierarchical statistical model based on his/her knowledge of the domain. By studying the summaries corresponding to the highest ranked nodes, the user may gradually recognize the factors that contribute to the observed variance. As expectations are adjusted, the variance may be reduced closer and closer to zero. Unlike traditional hierarchical statistical models, our approach allows multiple paths through the hierarchy.

Table 11
Top ranked nodes for login data after run 2

USER	LOGINTIME	Variance
Any	WDWE	0.0177
Any	AcademicYearAndTerm	0.0020
Any	AcademicYear	0.0015
Any	YYYY	0.0015
Any	FiscalYearAndQuarter	0.0013
Any	WeekdayName	0.0011
Any	FiscalYear	0.0011
Group	FiscalYear	0.0008

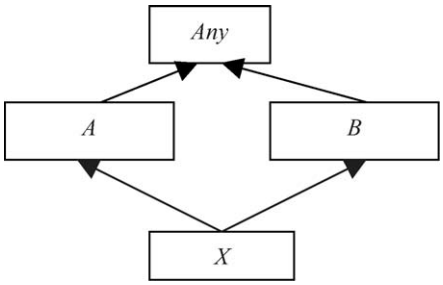

Fig. 8. DGG for synthetic data sets.

4.2. *Efficiency*

To measure the efficiency of the propagation in subgraphs created by the heuristic method proposed in Section 3, we first experimented on synthetic data sets, and then on the Saskatchewan weather data set.

Since the propagation time is directly proportional to the number of the records scanned in GenSpace graphs [13], we report propagation time in thousands of records scanned in our experiments. The advantage is that this measure is independent of the detailed implementation and the computers on which we ran the program.

We generated a set of tables with sizes ranging from 40 to 200 K. All tables have four attributes, a_1, a_2, a_3, and a_4. The possible values for these attributes are integers. All the values in the table are generated randomly. For simplicity, we give all attributes identical ExGen graphs, as shown in Fig. 8. We did two series of experiments to test the effect of cardinality and depth below which the nodes are marked as uninteresting.

Scalability. In the first series, we compared the time required using the propagation path produced by the proposed method to the time required using a path consisting of only potentially interesting nodes, as a function of the sizes of data sets. We set the number of sections in nodes A, B, and X to 5, 30, and 50, respectively, and varied the size of data sets from 40 to 200 K with an increment of 40 K. For this series, we marked the nodes under level four as uninteresting, which resulted in 162 uninteresting nodes and 94 potentially

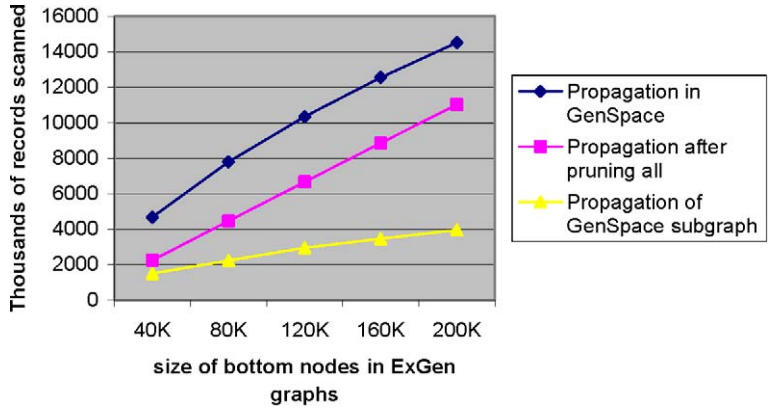

Fig. 9. Time cost for bottom nodes with different sizes.

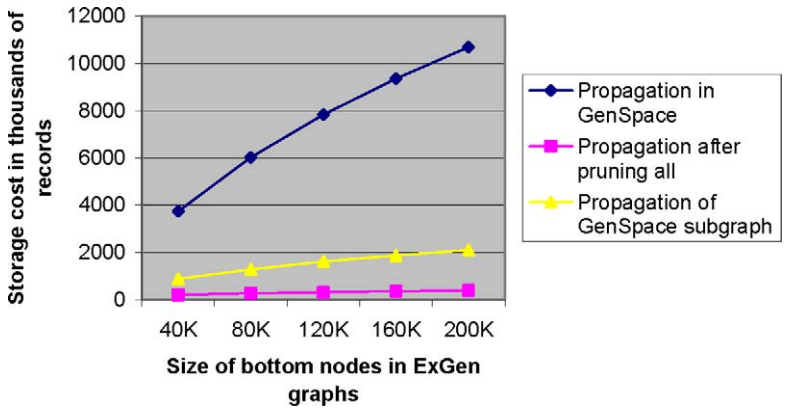

Fig. 10. Storage cost for bottom nodes with different sizes.

interesting nodes. Figs. 9 and 10 show the propagation cost and storage cost for bottom nodes with different sizes.

Depth of uninteresting nodes. In the second series, we marked nodes at varying depths as uninteresting. We set the size of A, B, and X to 8, 8, and 50, respectively. The size of the data set is 200 K. Figs. 11 and 12 compare the time and storage costs, respectively. As we anticipated, when we marked nodes below very low levels as uninteresting, the time savings are limited, because no nodes or only a few nodes are available for improving the propagation time, but when we marked all nodes below level three or a higher level as uninteresting, the time savings are significant.

For Saskatchewan weather data, we assumed two scenarios for the efficiency experiments. First, we assumed that all the nodes with depth less than or equal to four in the GenSpace graph are not interesting. In this case, 165 out of 560 nodes are uninteresting. In second scenario, we assumed that all nodes with specific date and specific temperature

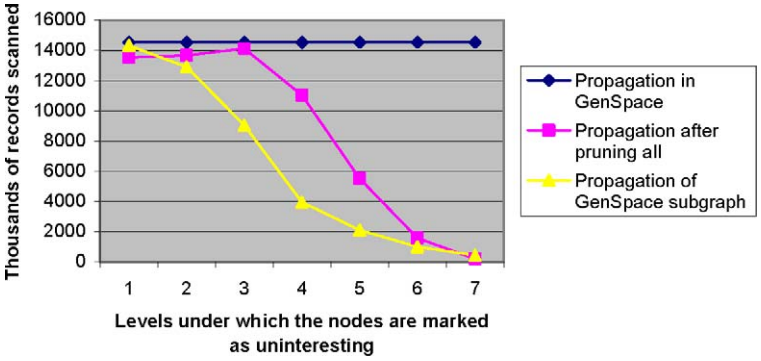

Fig. 11. Propagation time with different levels of uninteresting nodes.

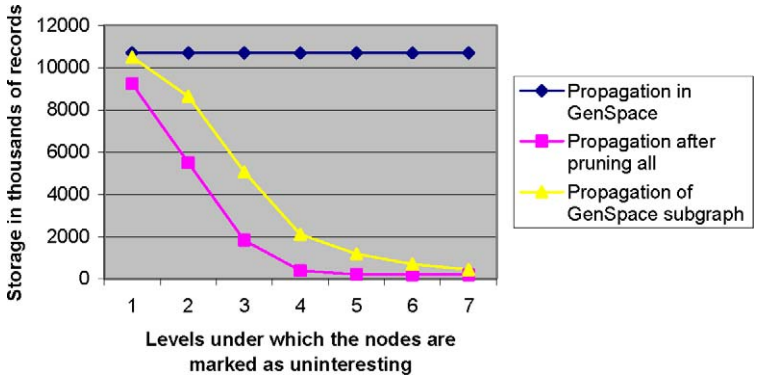

Fig. 12. Storage with different levels of uninteresting nodes.

values are not interesting. In this case, 200 nodes are uninteresting. Fig. 13(a) shows that the storage space (in thousands of records) for uninteresting nodes and the storage for preserved uninteresting nodes using heuristics for the first scenario, with the size of the bottom nodes ranging from 40 and 200 K. Fig. 13(b) shows the scanning costs for the GenSpace graph with and without pruning. Fig. 14 shows the corresponding trends for the second scenario. In these two scenarios, both storage and scanning costs are significantly reduced when we use our pruning strategy.

5. Conclusion

We have outlined an approach to summarization, a type of data mining that aggregates data in a variety of ways. Our approach is based on GenSpace graphs, and it is well suited for domains where calendar and geospatial attributes play a crucial role due to the complexity of background knowledge about these types of attributes. We specified the components of an expected distribution domain generalization graph suitable for a calendar attribute. Because of the complexity of the calendar DGG, it is useful to specify distributions of

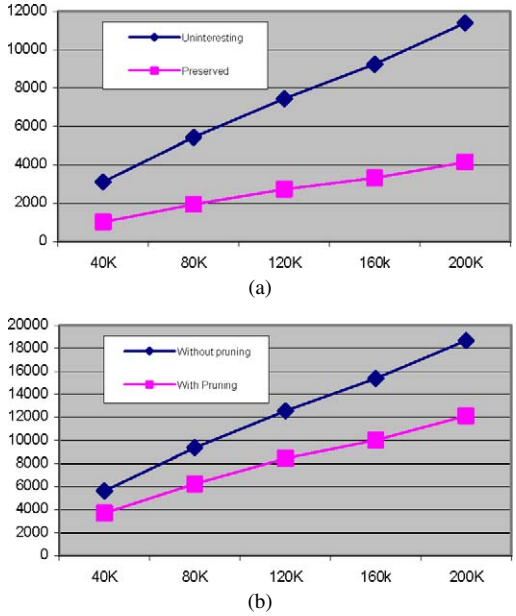

Fig. 13. Scenario 1. (a) Storage of uninteresting nodes versus storage of preserved nodes. (b) Scanning cost of original graph versus pruned graph.

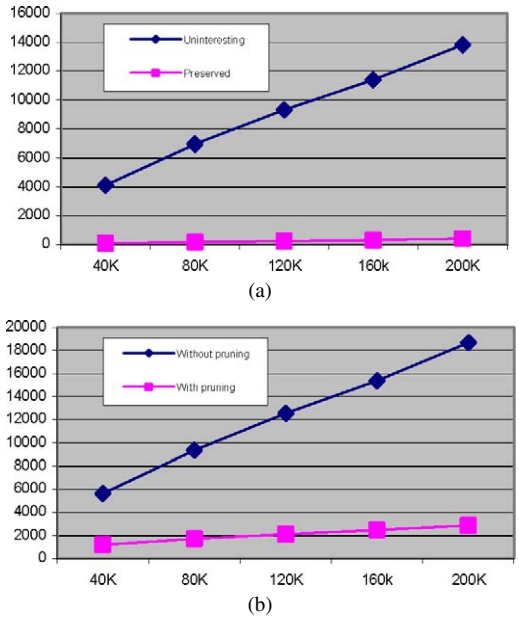

Fig. 14. Scenario 2. (a) Storage of uninteresting nodes versus storage of preserved nodes. (b) Scanning cost of original graph versus pruned graph.

values at various nodes and explore the consequences of these distributions on the interestingness ratings of various summaries of the same data. When applied to weather data and login data, using successive models of the user's expectations in each case, our approach conveniently identified several summaries that illustrated interesting facets of the data. We also addressed the efficiency issues by materializing appropriate nodes and identifying efficient propagation paths to reduce the storage and time costs.

Acknowledgements

We would like to thank Kathleen Hornsby and the anonymous reviewers for their constructive suggestions for improving this paper. This work was supported by the Institute for Robotics and Intelligence Systems (IRIS) Networks of Centers of Excellence of the government of Canada, PRECARN Associates Inc., a Natural Sciences and Engineering Research Council (NSERC) Research grant, Rogers Cablesystems, and the Faculty of Science, University of Regina.

References

[1] A. Andrusiewicz, M.E. Orlowska, On granularity factors that affect data mining, in: Eighth Internat. Database Workshop, Data Mining, Data Warehousing and Client/Server Databases, Hong Kong, 1997.
[2] C. Antunes, A. Oliveira, Temporal data mining: An overview, in: KDD 2001 Workshop on Temporal Data Mining, San Francisco, 2001.
[3] S.D. Bay, M.J. Pazzani, Detecting group differences: Mining contrast sets, Data Mining and Knowledge Discovery 5 (3) (2001) 213–246.
[4] E. Bertino, E. Ferrari, G. Guerrini, I. Merlo, Navigating through multiple temporal granularity objects, in: Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning (TIME'01), Cividale del Friuli, Italy, 2001, pp. 147–155.
[5] C. Bettini, S. Jajodia, X.S. Wang, Time Granularities in Databases, Data Mining, and Temporal Reasoning, Springer, Berlin, 2000.
[6] Y. Cai, N. Cercone, J. Han, Attribute-oriented induction in relational databases, in: G. Piatetsky-Shapiro, W.J. Frawley (Eds.), Knowledge Discovery in Databases, AAAI Press, 1991, pp. 213–228.
[7] C. Combi, F. Pinciroli, G. Pozzi, Managing time granularity of narrative clinical information: the temporal data model TIME-NESIS, in: Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME'96), Key West, FL, 1996, pp. 88–93.
[8] http://interactive.usask.ca/skinteractive/modules/environment/ecoregions.
[9] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery: An overview, in: U.M. Fayyad, G. Piatetsky-Shapiro, R. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 1–34.
[10] I. Goralwalla, Y. Leontiev, M.T. Özsu, D. Szafron, C. Combi, Temporal granularity: Completing the puzzle, J. Intelligent Inform. Syst. 16 (1) (2001) 41–63.
[11] H.J. Hamilton, L. Geng, L. Findlater, D.J. Randall, Spatio-temporal data mining with expected distribution domain generalization graphs, in: Proceedings of the 10th Symposium on Temporal Representation and Reasoning/International Conference on Temporal Logic (TIME-ICTL 2003), IEEE CS Press, Cairns, 2003, pp. 181–191.
[12] H.J. Hamilton, R.J. Hilderman, N. Cercone, Attribute-oriented induction using domain generalization graphs, in: Proceedings of the Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96), Toulouse, France, 1996, pp. 246–253.

[13] V. Harinarayan, A. Rajaraman, J.D. Ullman, Implementing data cubes efficiently, in: Proceedings of ACM SIGMOD'96, Montreal, Canada, 1996, pp. 205–216.
[14] R.J. Hilderman, H.J. Hamilton, Heuristic measures of interestingness, in: Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases, Prague, 1999, pp. 232–241.
[15] R.J. Hilderman, H.J. Hamilton, Knowledge Discovery and Measures of Interest, Kluwer Academic, Dordrecht, 2001.
[16] R.J. Hilderman, H.J. Hamilton, N. Cercone, Data mining in large databases using domain generalization graphs, J. Intelligent Inform. Syst. 13 (1999) 195–234.
[17] S. Kaufmann, Data visualization techniques for knowledge discovery based on domain generalization, M.Sc. Thesis, Department of Computer Science, University of Regina, Regina, Canada, 2002.
[18] C.H. Lee, C.R. Lin, M.S. Chen, On mining general temporal association rules in a publication database, in: Proceedings of the First IEEE International Conference on Data Mining, San Jose, CA, 2001, pp. 337–344.
[19] Y. Li, P. Ning, X.S. Wang, S. Jajodia, Discovering calendar-based temporal association rules, in: Proceedings of the Eighth International Symposium on Temporal Representation and Reasoning (TIME'01), Cividale del Friuli, Italy, 2001, pp. 111–118.
[20] C.P. Rainsford, J.F. Roddick, Adding temporal semantics to association rules, in: Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases, Prague, 1999, pp. 504–509.
[21] D.J. Randall, H.J. Hamilton, R.J. Hilderman, Generalization for calendar attributes using domain generalization graphs, in: Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning (TIME'98), Sanibel Island, FL, 1998, pp. 177–184.
[22] D.J. Randall, Temporal generalization in databases using domain generalization graphs, M.Sc. Thesis, Department of Computer Science, University of Regina, Regina, Canada, 2002.
[23] J. Zytkow, From contingency tables to various forms of knowledge in databases, in: U.M. Fayyad, G. Piatetsky-Shapiro, R. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 329–349.